Scheduling Algorithm

จัดทำโดย
นายสุริยา เตชะลือ 600610790

เสนอ
รศ.ดร.นริศรา เอี่ยมคณิตชาติ

สารบัญ

Concept & code of 3 algorithm

First Come First Search

```
def FCFS(FCFSTime, processList, numProcess):
    temp = 0
    for x in range(len(processList)):
        FCFSTime += temp
        temp += processList[x]
    FCFSTime = FCFSTime/numProcess
    return FCFSTime
```

ทำการประมวลผลตัวแรกที่เข้ามา Process ก่อนจากนั้นเรียงลำดับถัดไป waiting time จะเพิ่มขึ้นตามจำนวนที่ process ก่อนหน้าใช้ในการรอ และ average waiting time เท่ากับ waiting time ของ process ทั้งหมด หารด้วย จำนวน process ทั้งหมด ฟังก์ชัน FCFS คืนค่า average waiting time ของ First Come First Search

Shortest Job First

```
def SJF(SJFTime, tempprocessList, numProcess):
    temp = 0
    for x in range(len(tempprocessList)):
        SJFTime += temp
        temp += min(tempprocessList)
        tempprocessList.remove(min(tempprocessList))
    SJFTime = SJFTime/numProcess
    return SJFTime
```

เนื่องจากไม่มีการคิด arrival time ดังนั้น process ทุกตัวจึงมีลำดับการประมวลที่เท่ากัน เพราะฉะนั้นจึงให้ process ที่มีเวลาน้อยที่สุดมาประมวลผลก่อน จึงดึงค่าที่น้อยที่สุดออกจาก list process เพื่อมาประมวลผล waiting time และ average waiting time เท่ากับ waiting time ของ process ทั้งหมด หารด้วย จำนวน process ทั้งหมด ฟังก์ชัน SJF คืนค่า average waiting time ของ Shortest Job First

Round Robin

```
def RR(RRTime, temp2processList, numProcess):
    countTime = 0
    endProcess = [0] * numProcess
    timeQuantum = 8
    point = 0
    while len(temp2processList) > 0:
        point = point % (len(temp2processList))
        RRTime += (countTime - endProcess[point])
        if temp2processList[point] > timeQuantum:
            temp2processList[point] -= timeQuantum
            countTime += timeQuantum
            endProcess[point] = countTime
        elif temp2processList[point] <= timeQuantum:
            countTime += temp2processList[point]
            temp2processList.remove(temp2processList[point])
            endProcess.remove(endProcess[point])
            point -= 1
        point += 1
    RRTime = RRTime/numProcess
    return RRTime
```

การจัดคิวให้แต่ละ process เข้าไปประมวลผลตามลำดับ แต่มี Time Quantum ในการกำหนดเวลาที่แต่ละ process เข้าไปทำงานได้ โดยถ้าหมด Time Quantum ก็จะไปทำยัง process ถัดไป ค่าที่ได้จากการเรียกใช้ฟังก์ชันคือ average waiting time ของ Round Robin

Experimental results

Data set

Number of Process : 10

number of process	case 1	case 2	case 3	case 4	case 5	case 6	case 8	case 9	case 10
2 - 8 ms	16	30	28	22	18	21	30	26	25
20 - 30 ms	7	15	8	14	8	13	20	9	17
35 - 40 ms	2	9	8	8	4	5	6	8	5
total	25	54	44	44	30	39	56	43	47

percent of process	case 1	case 2	case 3	case 4	case 5	case 6	case 8	case 9	case 10
2 - 8 ms	64	55.55556	63.63636	50	60	53.84615	53.57143	60.46512	53.19149
20 - 30 ms	28	27.77778	18.18182	31.81818	26.66667	33.33333	35.71429	20.93023	36.17021
35 - 40 ms	8	16.66667	18.18182	18.18182	13.33333	12.82051	10.71429	18.60465	10.6383

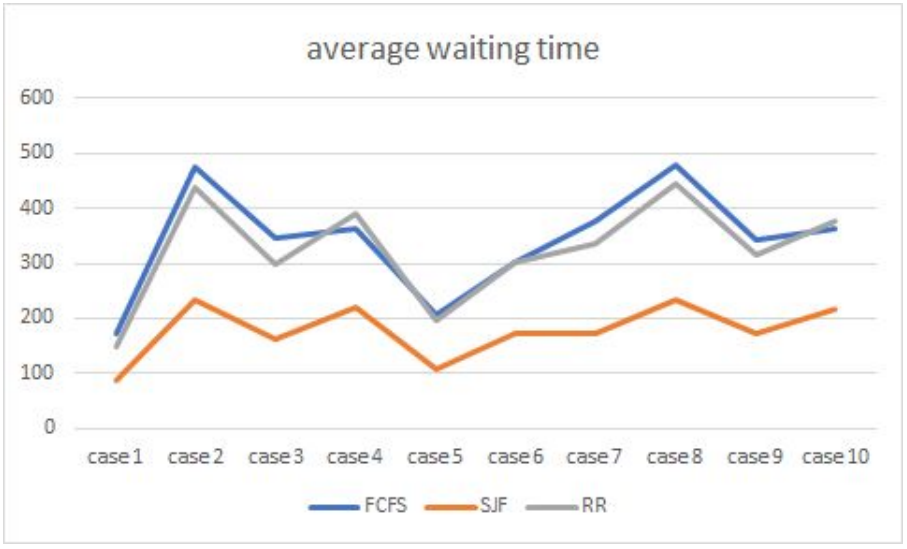
average waiting time	case 1	case 2	case 3	case 4	case 5	case 6	case 8	case 9	case 10
FCFS	172.68	476.1852	345.0455	363.75	205.9667	300.2564	478.8393	341.2558	362.0638
SJF	86.68	235	163.2045	218.8182	108.9667	171.1795	233.1607	173.8372	216.9787
RR	149	438.2407	297.5	390.9091	197.5667	303.4615	445.5179	315.8605	375.8085

Data list										
No. case	case 1	case 2	case 3	case 4	case 5	case 6	case 7	case 8	case 9	case 10
Sum time process	326	876	653	773	435	626	675	863	682	756
	22	37	6	7	36	40	6	20	3	7
	36	3	8	36	21	3	26	2	25	6
	6	26	8	4	25	4	20	36	30	7
	8	7	6	20	8	29	29	28	6	2
	4	8	38	3	8	23	38	29	40	7
	39	40	8	4	2	23	3	7	7	23
	22	27	27	24	5	6	27	40	4	21
	8	7	4	3	5	4	40	25	25	30
	20	5	35	4	8	35	22	30	37	38
	6	40	38	30	4	8	36	29	28	8
	3	5	3	36	8	2	35	6	5	23
	6	8	2	7	21	4	22	6	2	7
	4	27	7	3	21	30	25	4	3	5
	5	4	36	38	6	29	4	27	5	8
	5	8	7	30	3	4	3	5	7	7
	21	39	36	36	25	5	3	28	36	37
	8	23	30	21	5	38	6	2	8	6
	24	20	7	4	39	3	2	20	39	28
	7	39	5	27	5	25	4	2	7	2
	5	26	6	23	30	3	23	20	6	8
	26	21	3	8	27	4	23	39	25	28
	5	5	30	4	38	7	2	2	4	6

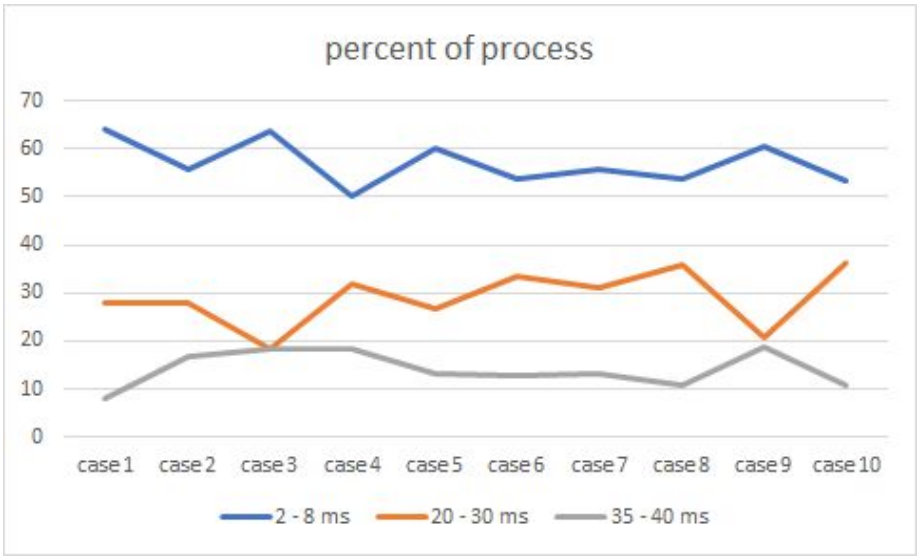
	8	2	6	36	3	8	39	6	3	22
	2	39	6	4	5	29	5	20	25	26
	26	23	36	7	4	8	7	38	36	23
		38	25	8	5	26	3	24	7	8
		6	8	30	2	5	2	24	8	6
		3	40	22	2	7	4	23	27	4
		7	3	27	29	20	7	3	6	37
		26	21	7	35	35	4	26	8	30
		29	29	2		8	6	7	30	39
		27	40	38		30	4	7	4	20
		5	21	8		7	20	8	39	24
		5	4	29		20	8	5	2	40
		5	2	5		23	4	4	39	6
		8	5	37		22	28	2	4	4
		36	5	7		4	23	4	3	22
		2	3	28		7	3	39	3	27
		3	27	8		38	27	23	3	20
		4	2	23			37	6	39	30
		4	7	6			4	4	8	6
		3	6	30			2	5	8	7
		8	3	4			30	24	28	3
		37	4	35			3	39		8
		24					6	4		7
		28						7		2
		22						3		21
		6						25		

		8						4		
		4						6		
		3						3		
		26						5		
		3						7		
		7						23		
								4		
								24		

Results



กราฟเปรียบเทียบ average waiting time ระหว่าง First come first search, Shortest job first และ Round robin



กราฟเปรียบเทียบอัตราส่วนของเวลาในแต่ละ process ระหว่าง First come first search,Shortest job first และRound robin

Conclusion

จากข้อมูล Data set ข้างต้นที่ได้นำมาทดลองรันผ่านฟังก์ชัน FCFS, SJF และ RR และลัพธ์ของเวลาที่ได้ เมื่อนำมาสร้างกราฟ Average waiting time แสดงให้เห็นว่า อัลกอริทึมในการจัดเวลาให้แต่ละ Process เข้าไปทำงานยัง CPU อัลกอริทึม Shortest job first ดีที่สุดเพราะให้ process ที่ใช้เวลาในการประมวลผลน้อยที่สุดทำงานก่อน ทำให้ average waiting time น้อยที่สุด รองลงมาคือ Round robin ที่มีการกำหนด Time Quantum เท่ากับ 8 ms ทำให้การทำงานในบางช่วงดีกว่า First come first search แต่อาจจะมีบางช่วงที่ทำงานช้ากว่า เนื่องจากมีอัตราส่วนของ process ที่น้อยกว่า Time Quantum ในอัตราส่วนที่น้อย และสุดท้ายอัลกอริทึม First come first search ที่ให้ process ที่มาก่อนได้ทำงานก่อน

อ้างอิง

https://github.com/fsuriya/OS